VIETNAM INTERNATIONAL UNIVERSITY – HO CHI MINH CITY

INTERNATIONAL UNIVERSITY

**WEB APPLICATION DEVELOPMENT PROJECT**

**FURNITURE SALES WEBSITE**

BY

Trịnh Quang Anh – Student ID: ITCSIU21164

Advisor: Dr. Nguyen Van Sinh

A report submitted to the School of Computer Science and Engineering

in partial fulfillment of the requirements for the Final Project

in Web Application Development course - Fall 2024

Ho Chi Minh city, Vietnam, 2024

Contents

[INTRODUCTION 3](#_Toc185629981)

[SYSTEM DIAGRAM 3](#_Toc185629982)

[DETAIL DESCRIPTION OF FUNCTIONS 4](#_Toc185629983)

[DEVELOPMENT PROCESS 5](#_Toc185629984)

[DATABASE 6](#_Toc185629985)

[REQUIREMENT ANALYSIS AND DESIGN 7](#_Toc185629986)

[REQUIREMENT ANALYSIS 7](#_Toc185629987)

[FUNCTIONAL REQUIREMENTS 20](#_Toc185629988)

[NON-FUNCTIONAL REQUIREMENTS 30](#_Toc185629989)

[Design 31](#_Toc185629990)

[Django Framework 41](#_Toc185629991)

[MVT model 41](#_Toc185629992)

[MVT work flow 41](#_Toc185629993)

[django.core.mail module 42](#_Toc185629994)

[IMPLEMENTATION 42](#_Toc185629995)

[DISCUSSION AND CONCLUSION 45](#_Toc185629996)

[REFERENCES 46](#_Toc185629997)

# INTRODUCTION

With a growing population and increasing population density, the demand for furniture also increases accordingly. Especially in big cities, the need to "move from house to apartment" is increasing. Besides, online shopping is gradually replacing buying directly from stores. From there, creating furniture buying and selling websites is necessary for the market. Furniture Sale Website was born to meet consumer tastes.

In this project, our team used HTML, CSS, JavaScript to create the interface, and used Django to manage the BackEnd

After implementing the project, the team achieved results such as: building a simple interface for the Furniture Sale Website, the function of adding products to the shopping cart, the function of registering and logging in for users, and the search function. search for products.

## SYSTEM DIAGRAM

The HomeClick website system includes the functions listed in the image below:

A diagram of a product

Description automatically generated

Figure 1: System Diagram

## DETAIL DESCRIPTION OF FUNCTIONS

1. Product Management

* Add New Product: administrator adds new product to the system.
* Delete Product: delete the product from the system.
* Update Product Information: update existing product information.
* Product Search: users search for products according to different criteria.

1. Order Management

* Add New Order: user creates new order when purchasing, new order is created when user clicks on the "Add to cart" button for each product.
* Update Order: admin updates order status.
* Delete order: administrator deletes order from the system.

1. User Management

* Registration: user creates a new account, user account information is saved in the database.
* Log in: if you already have an account, you must log in to make a purchase.
* Update address, update phone number: on the User Profile page, it allows updating the user's address and phone number to save delivery information.
* Delete customer: administrator deletes customer account in database.

1. Cart Management

* Add products, reduce products: users are allowed to increase or decrease products in the shopping cart. If the number of products decreases to 0, the product will be removed from the shopping cart.
* Count the number of products in the cart: based on the number of products in the cart and display the number of products.
* Calculate the total amount of the order: calculate the total amount of all remaining products in the shopping cart.

## DEVELOPMENT PROCESS

1. Requirement Analysis

* Gather requirements from stakeholders, including product categories, target audience, and website functionality.
* Document functional and non-functional requirements.
* Define project scope, objectives, and deliverables.

1. Planning

* Create a project roadmap and timeline.
* Allocate resources, including team members and tools.
* Define milestones and key deliverables.
* Prepare a risk management plan.

1. Design Phase

* UI/UX Design:
* Develop wireframes and mockups for the website.
* Ensure responsive design for desktop, tablet, and mobile devices.
* Incorporate branding guidelines and user-friendly navigation.
* System Architecture Design:
* Design the database schema (e.g., products, users, orders).
* Define the architecture (e.g., client-server model, APIs).

1. Development Phase

* Frontend Development
* Implement responsive design using HTML, CSS, and JavaScript.
* Build components for product display, search filters, shopping cart, and checkout.
* Ensure seamless user interaction through dynamic features.
* Backend Development
* Set up the server and database
* Develop APIs for product management, user authentication, and order processing.
* Integrate a secure payment gateway

1. Testing Phase

* Unit Testing: Test individual components (e.g., product listing, checkout process).
* Integration Testing: Ensure seamless communication between frontend, backend, and external APIs.
* System Testing: Validate the complete system functionality.
* User Acceptance Testing (UAT): Collect feedback from end-users to ensure the website meets expectations.
* Performance Testing: Ensure the website can handle high traffic and large data loads.

1. Development Phase

* Prepare the production environment.
* Deploy the website to a live server
* Conduct a final round of testing on the live site.

## DATABASE

The image below is the database used to build the website

A screenshot of a computer

Description automatically generated

Figure 2: Database of Furniture Sales Website

* Customer table contains personal information of customers. This information includes the customer's name, email, phone number and address.
* Product table stores information about available products on the website. Product information includes name, price, product code, type and image.
* Order table stores information about customer orders, such as order code, ordering customer, order date, completion status and transaction code.
* OrderItem table links between the Order table and the Product table, containing information about the products in each order, including quantity and date added to cart.
* ShippingAddress table stores information about the customer's shipping address, including detailed address, city, state, contact phone number, and date the address was added.
* Article table contains information about articles, including name, posting date, and image.

# REQUIREMENT ANALYSIS AND DESIGN

This section briefly introduces the requirement analysis and design process. This version is created to provide the path for future implementation of the project. Based on this requirements specification, we will implement each function including all of the conditions as well as functional and non-functional requirements supplied by the customers. During our implementation, we will constantly revise and update the newer version so that we can always keep track of the progress of the project.

## REQUIREMENT ANALYSIS

**Use Case Diagram**

A diagram of a product

Description automatically generated

Figure 3: Use Case Diagram

**Use Case 1:**

Name: Log in the system

Identifier UC1

Inputs:

* Gmail
* Password

Outputs:

* The home page with user’s authorization [If success]
* The login page [If fail]

Basic Course

|  |  |
| --- | --- |
| Actor: User (Customer/Admin) | System |
| 1. Open the login page | * 1. Display the login page |
| 1. Enter gmail and password |  |
| 1. Submit | * 1. Check the user’s info   2. If success, return the home page   3. Else return the login page |

Pre-conditions:

* User has an registered account of online store that is created earlier (ID and password)

Post-conditions:

* None

User story: As an online shop visitor, I want to log in the shop system so that I can use functions of the shop like buy products, gain the promotion, discuss about the products …

**Use Case 2:**

Name: Add product into cart

Identifier UC2

Inputs:

* Product Name
* Product Description
* Product Price
* Product Images

Outputs:

* Confirmation message: "Product added successfully."
* Updated product catalog including the new product.
* Log entry for the new product addition.

Basic Course:

|  |  |
| --- | --- |
| Actor: User | System |
| 1. The user logs into the system. | * 1. Validates the input data for completeness and correctness. |
| 1. The user chooses the product they want to buy |  |
| 1. The user clicks on the "Add Product" button. | * 1. Stores the product details in the database. |
| 1. The user clicks "Submit" to add the product. | * 1. Updates the product in cart   2. Displays a confirmation message to the user |

Pre-conditions:

* The product must exist in the database.
* The product must have sufficient stock available.
* The user must be logged in or have an active session (for tracking cart contents).

Post-conditions:

* The selected product is successfully added to the user's cart.
* The cart is updated in the database, and the total cart value is recalculated.
* The user sees the updated cart with the added product.

User story: I want to add products to my shopping cart. So that I can review them and proceed to checkout when ready to purchase.

**Use Case 3:**

Name: Delete product from cart

Identifier UC3

Inputs:

* Product ID
* Quantity of the product
* User ID

Outpus:

* Confirmation message: "Product successfully added to cart."
* Updated cart displaying the added product, quantity, and total price.

Basic Course:

|  |  |
| --- | --- |
| Actor: User | System |
| 1. The user navigates to their shopping cart. |  |
| 1. The user identifies the product they want to remove. | * 1. Retrieves the user's current cart details from the database.   2. Identifies the product to be removed using the Product ID. |
| 1. The user clicks on the "Remove" or "Delete" button next to the product. | * 1. Removes the product from the cart in the database.   2. Recalculates the cart's total value after removing the product.   3. Updates the user's cart to reflect the changes.   4. Displays a confirmation message to the user. |

Pre-conditions:

* The user must be logged in or have an active session (to track the cart).
* The cart must contain at least one product.
* The product to be deleted must exist in the user's cart.

Post-conditions:

* The selected product is successfully removed from the user's cart.
* The cart in the database is updated to reflect the removal.
* The user sees the updated cart with the remaining items and recalculated total.

User story: I want to remove products from my shopping cart. So that I can adjust my selections before proceeding to checkout.

**Use Case 4:**

Name: Update product from cart

Identifier UC4

Inputs:

* Product ID (the product to be updated)
* New Quantity (the updated quantity of the product)
* User ID

Outputs:

* Confirmation message: "Product quantity updated successfully."
* Updated cart displaying the product with the new quantity and recalculated total price.

Basic Course:

|  |  |
| --- | --- |
| Actor: User | System |
| 1. The user navigates to their shopping cart. |  |
| 1. The user identifies the product they want to update. | * 1. Retrieves the user's current cart details from the database.   2. Validates the new quantity (e.g., ensures it does not exceed available stock or is not less than 1). |
| 1. The user modifies the quantity using a quantity input field or +/- buttons. | * 1. Updates the product quantity in the cart stored in the database.   2. Recalculates the cart's total value based on the updated quantity. |
| 1. The user clicks the "Update" button to confirm the changes. | * 1. Updates the user's cart to reflect the new quantity and total price.   2. Displays a confirmation message to the user. |

Pre-conditions:

* The user must be logged in or have an active session (to track the cart).
* The cart must contain the product to be updated.
* The product must have sufficient stock to accommodate the updated quantity.

Post-conditions:

* The product's quantity in the cart is successfully updated.
* The cart in the database reflects the changes.
* The user sees the updated cart with the new quantity and recalculated total.

User story: I want to update the quantity of a product in my shopping cart. So that I can adjust my purchase according to my needs before proceeding to checkout.

**Use Case 5:**

Name: Update product in website

Identifier UC5

Inputs:

* Product ID (the product to be updated)
* Updated Product Details:
* Name
* Category
* Description
* Price
* Quantity in Stock
* Dimensions and Material Details
* Discount
* Updated Product Images

Outputs:

* Confirmation message: "Product information updated successfully."
* Updated product information displayed on the website.

Basic Course:

|  |  |
| --- | --- |
| Actor: Admin | System |
| 1. The admin logs into the system. |  |
| 1. The admin navigates to the "Product Management" section. |  |
| 1. The admin selects the product to be updated from the product list. |  |
| 1. The admin modifies the necessary product details (e.g., name, price, quantity). | * 1. Validates the updated product information for completeness and correctness. |
| 1. The admin clicks the "Update" button to save the changes. | * 1. Updates the product details in the database.   2. Displays a confirmation message to the admin. |

Pre-conditions:

* The admin must be authenticated and logged into the system.
* The product to be updated must exist in the database.

Post-conditions:

* The product's details in the database are successfully updated.
* The updated product information is displayed on the website.
* Customers can view the updated product information in the product catalog.

User story: I want to update the details of a product on the website. So that customers can view the latest and most accurate information about the product.

**Use Case 6:**

Name: Find product

Identifier UC6

Inputs:

* Search Keywords (e.g., product name, category, or brand)
* Filters (optional):
* Price range
* Product category
* Availability (in stock or out of stock)
* Ratings and reviews

Outputs:

* List of products matching the search criteria, including:
* Product name
* Thumbnail image
* Price
* Short description
* Availability status

Basic Course:

|  |  |
| --- | --- |
| Actor: User | System |
| 1. The user navigates to the website's homepage or product search page. |  |
| 1. The user enters a search keyword in the search bar or selects specific filters. | * 1. Receives the search input and selected filters from the user.   2. Queries the database for products matching the search criteria. |
| 1. The user clicks the "Search" button or presses Enter. | * 1. Displays the search results to the user, including relevant details for each product. |

Pre-conditions:

* The product catalog must exist in the database.
* The search functionality must be active and connected to the database.

Post-conditions:

* The system displays a list of products that match the search criteria or a "No results found" message if no matches are found.
* The user can browse the search results and select a product for more details.

User story: I want to search for products on the website. So that I can quickly find the items I am interested in purchasing.

**Use Case 7:**

Name: Register account

Identifier UC7

Inputs:

* Full Name
* Email Address
* Password (and Confirm Password)
* Phone Number (optional)
* Address (optional)
* Agreement to Terms and Conditions (checkbox)

Outputs:

* Confirmation message: "Account successfully created."
* Email notification (e.g., account activation or welcome email).

Basic Course:

|  |  |
| --- | --- |
| Actor: User | System |
| 1. The user navigates to the "Register" or "Sign Up" page. |  |
| 1. The user enters their details (name, email, password, etc.). | * 1. Validates the user's input (e.g., checks for required fields, valid email format, password strength).   2. Checks if the email address is already associated with an existing account.   3. Creates a new user account in the database if validation passes. |
| 1. The user agrees to the terms and conditions by checking the appropriate box. | * 1. Sends a confirmation or activation email to the user. |
| 1. The user clicks the "Register" or "Sign Up" button. | * 1. Displays a success message to the user on the website. |

Pre-conditions:

* The user is not already registered with the provided email.
* The registration form is accessible on the website.
* The system must be connected to the database and email server.

Post-conditions:

* A new user account is created and stored in the database.
* The user receives a confirmation or activation email.
* The user can log in to the system using their new account credentials.

User story: I want to register an account on the website. So that I can access personalized features and make purchases seamlessly.

**Use Case 8:**

Name: Update information

Identifier UC8

Inputs:

* User ID (from logged-in session)
* Updated User Information:
* Full Name
* Email Address
* Phone Number
* Address
* Profile Picture (optional)
* Password (if applicable, with confirm password field)

Outputs:

* Confirmation message: "Your information has been successfully updated."
* Updated user profile displayed on the user's account page.

Basic Course:

|  |  |
| --- | --- |
| Actor: User | System |
| 1. The user logs into their account. |  |
| 1. The user navigates to the "Account Settings" or "Profile" page. | * 1. Authenticates the user's session to ensure they are authorized to make updates. |
| 1. The user edits the desired fields (e.g., name, phone number, address). | * 1. Validates the updated information (e.g., checks email format, phone number validity). |
| 1. The user clicks the "Save" or "Update" button to apply the changes. | * 1. Updates the user's information in the database.   2. Displays the updated information on the user's profile page.   3. Shows a success message confirming the changes. |

Pre-conditions:

* The user must be logged into their account.
* The "Update Information" feature must be available and connected to the database.

Post-conditions:

* The user's updated information is saved in the database.
* The updated profile information is displayed on the website.
* A success confirmation is shown to the user.

User story: I want to update my personal information on my account. So that my profile remains accurate and up to date.

**Use Case 9:**

Name: Add order

Identifier UC9

Inputs:

* User ID
* Order Details
* Product(s) (Product ID and Quantity)
* Shipping Address
* Payment Method (Cash on Delivery)
* Contact Information (Name, Phone Number, Email)

Outputs:

* Confirmation message: "Order placed successfully."
* Order summary, including:
* Order ID
* Product details (name, quantity, price)
* Shipping details
* Total cost
* Estimated delivery date

Basic Course:

|  |  |
| --- | --- |
| Actor: User | System |
| 1. The user logs into their account | * 1. Authenticates the user's session |
| 1. The user reviews their shopping cart and proceeds to checkout. | * 1. Validates the shopping cart details (e.g., ensures product availability and valid quantities).   2. Validates the shipping and payment information. |
| 1. The user enters or selects a shipping address. | * 1. Creates a new order in the database, associating it with the user.   2. Deducts the ordered quantities from the product inventory.   3. Generates an order summary and displays it to the user. |
| 1. The user selects a payment method and provides payment details if required. |  |
| 1. The user confirms the order by clicking the "Place Order" button. | * 1. Sends a confirmation email or notification to the user. |

Pre-conditions:

* The user must be logged in.
* The shopping cart must contain at least one product.
* The selected products must be in stock.
* Valid payment and shipping details must be provided.

Post-conditions:

* The new order is saved in the database with a unique Order ID.
* The ordered products' inventory is updated.
* The user receives an order confirmation and summary.

User story: I want to place an order for the products in my shopping cart. So that I can complete my purchase and receive the items at my specified address.

**Use case 10:**

Name: Delete Order

Identifier UC10

Inputs:

* User ID
* Order Details:
* Product(s) (Product ID and Quantity)
* Shipping Address
* Payment Method (e.g., Credit Card, PayPal, Cash on Delivery)
* Contact Information (Name, Phone Number, Email)

Outputs:

* Confirmation message: "Order placed successfully."
* Order summary, including:
* Order ID
* Product details (name, quantity, price)
* Shipping details
* Total cost
* Estimated delivery date

Basic Course:

|  |  |
| --- | --- |
| Actor: User | System |
| 1. The user logs into their account. | * 1. Authenticates the user's session |
| 1. The user reviews their shopping cart and proceeds to checkout. | * 1. Validates the shopping cart details   2. Validates the shipping and payment information. |
| 1. The user enters or selects a shipping address. | * 1. Creates a new order in the database, associating it with the user.   2. Deducts the ordered quantities from the product inventory. |
| 1. The user selects a payment method and provides payment details if required. |  |
| 1. The user confirms the order by clicking the "Place Order" button. | * 1. Generates an order summary and displays it to the user.   2. Sends a confirmation email or notification to the user. |

Pre-conditions:

* The user must be logged in.
* The shopping cart must contain at least one product.
* The selected products must be in stock.
* Valid payment and shipping details must be provided.

Post-conditions:

* The new order is saved in the database with a unique Order ID.
* The ordered products' inventory is updated.
* The user receives an order confirmation and summary.

User story: I want to place an order for the products in my shopping cart. So that I can complete my purchase and receive the items at my specified address.

**Use case 11:**

Name: Update Order

Identifier UC11

Inputs:

* User ID
* Order ID (the order to be updated)
* Updated Order Details
* Product(s) (add, remove, or modify quantities)
* Shipping Address
* Payment Method
* Order Notes (optional)

Outputs:

* Confirmation message: "Order updated successfully."
* Updated order summary, including:
* Order ID
* Updated product details (name, quantity, price)
* Updated shipping details
* Revised total cost

Basic Course:

|  |  |
| --- | --- |
| Actor: Users | System |
| 1. The user or admin logs into their account. | * 1. Authenticates the user's or admin's session. |
| 1. The user or admin navigates to the "Order Management" section. |  |
| 1. The user or admin selects the order they want to update. | * 1. Retrieves the order details based on the Order ID. |
| 1. The user or admin modifies the desired order details (e.g., changes quantity, updates shipping address, etc.). | * 1. Validates the updated details   2. Updates the order in the database with the new information.   3. Recalculates the total cost of the order |
| 1. The user or admin confirms the changes by clicking the "Update Order" button. | * 1. Displays the updated order summary to the user or admin.   2. Sends a notification or email to the user about the updated order. |

Pre-conditions:

* The user or admin must be logged in.
* The order to be updated must exist in the system.
* Any updated product quantities must be available in stock.
* The update must occur within an allowed time frame (if applicable, e.g., before shipping).

Post-conditions:

* The order is updated in the database with the revised details.
* The updated order summary is displayed to the user or admin.
* The inventory, if affected, is adjusted accordingly.

User story: I want to update my existing order. So that I can make changes to the products, shipping address, or payment details before the order is processed.

## FUNCTIONAL REQUIREMENTS

**Use Case 1: Log in the system**

1. **The scope of the work**

* This occurs in the sprint 5 in the process
* 4 tasks needed for this function
* 20 hours of effort is needed for this function

1. **The scope of the product:** This is the main part of user’s login functionality
2. **Functional and Data Requirements:**

* **Funtional Requirements**
* Shall display the log in so that user can input the information of account
* Shall achieve the information of the account that is matched with the ID that user logs in from the database
* Shall decrypt password of user account from the database
* Shall check the validity of user login account
* **Data Requirements**: The log in information (ID and password) must be valid

**Use Case 2: Add product into cart**

1. **The scope of the work:**

* Frontend: Develop a user-friendly interface for adding products to the cart.
* Backend: Implement logic for cart management, inventory validation, and updating cart data in the database.
* Testing: Ensure the functionality works as intended on all supported devices and browsers.

1. **The scope of the product:**

* Select a product from the product listing or detail page.
* Specify the desired quantity.
* Add the selected product to their shopping cart.
* View an updated cart summary showing the product, quantity, and total price.
* Inventory validation (products cannot be added if out of stock or beyond available quantity).
* Cart data is persisted for logged-in users and stored temporarily for guest users.

1. **Functional and Data Requirements:**

* **Funtional Requirements:**
* Users can select a product and specify a quantity.
* Users can click an "Add to Cart" button to add the product to their cart.
* Validate that the selected quantity is available in stock.
* Update the user's cart with the selected product and quantity.
* Recalculate and display the cart's total price.
* Provide confirmation that the product has been added successfully.
* **Data Requirements:**
* Inputs:
* Product ID: Unique identifier for the product being added.
* Quantity: Desired quantity of the product.
* User ID: Identifier for the logged-in user
* Outputs:
* Cart Data
* Cart Summary: Updated total price of the cart.
* Confirmation Message: Message to notify the user that the product was added successfully.
* Validation Rules:
* Verify product availability in inventory.
* Ensure the selected quantity is valid (e.g., greater than 0 and does not exceed available stock).

**Use Case 3: : Delete product from cart**

1. **The scope of the work:**

* Update the cart data dynamically in the system.
* Recalculate the total price of the remaining products in the cart.
* Ensure cart data persistence for logged-in users and temporary storage for guest users.

1. **The scope of the product:**

* Update the cart data dynamically in the system.
* Recalculate the total price of the remaining products in the cart.
* Ensure cart data persistence for logged-in users and temporary storage for guest users.

1. **Functional and Data Requirements:**

* **Functional Requirements:**
* Users can view their shopping cart.
* Users can click a "Delete" or "Remove" button next to a product to remove it from the cart.
* Validate the existence of the product in the cart.
* Remove the selected product from the cart (database for logged-in users, local storage for guests).
* Recalculate the total price of the cart.
* Update the cart summary to reflect the change.
* Display a confirmation message to the user.
* For logged-in users, update the cart in the database.
* For guest users, update the cart stored in local storage or cookies.
* **Data Requirements:**
* Inputs:
* Product ID: Unique identifier of the product to be deleted.
* User ID: Identifier for the logged-in user (if applicable).
* Outpus:
* Updated Cart Data
* Confirmation Message: "Product successfully removed from the cart."
* Validation Rules:
* Verify that the product exists in the user's cart.
* Ensure the cart is updated without errors.

**Use Case 4: Update product from cart**

1. **The scope of the work:**

* Frontend: Develop an intuitive user interface to allow users to edit product quantities in their cart.
* Backend: Implement logic to validate, update, and save the product quantity in the database or local storage.
* Testing: Validate that updates are applied correctly, with error handling for edge cases

1. **The scope of the product:**

* Valid input (quantity > 0 and does not exceed available stock).
* The cart data is updated dynamically and displayed to the user.
* Inventory consistency (product stock availability is validated).

1. **Functional and Data Requirements:**

* **Functional Requirements:**
* Users can view their cart and locate the product to be updated.
* Users can change the product quantity using an input field or +/- buttons.
* Users can confirm the update by clicking an "Update" button.
* Validate the new quantity (e.g., must be greater than 0 and within available stock).
* Update the product quantity in the cart
* Recalculate the cart subtotal for the updated product.
* Recalculate the total cart value.
* Display an updated cart summary to the user.
* Show a success message if the update is successful or an error message if validation fails.
* **Data Requirements:**
* Inputs:
* Product ID: Unique identifier of the product in the cart.
* New Quantity: Desired quantity entered by the user.
* User ID: Identifier for logged-in users (if applicable).
* Outputs:
* Updated Cart Data
* Confirmation Messages
* Validation rules:
* The updated quantity must be greater than 0.
* The updated quantity must not exceed the available stock in the inventory.

**Use Case 5: Update product in website**

1. **The scope of the work:**

* Frontend: Develop an intuitive interface for admins to update product details.
* Backend: Implement validation logic, update product data in the database, and manage image uploads.
* Testing: Ensure successful updates with no data loss and proper error handling.

1. **The scope of the product:**

* Validation of all product fields.
* Updates are consistent with the database and visible to customers.
* Admins receive confirmation of successful changes.

1. **Functional and Data Requirements:**

* **Functional Requirements:**
* Log into the admin panel.
* Navigate to the "Product Management" section.
* Select a product to update.
* Modify product details (e.g., name, description, price, stock, images).
* Click "Save" or "Update" to confirm changes.
* Authenticate the admin session.
* Retrieve the existing product details.
* Validate the updated inputs (e.g., numeric price, positive stock).
* Update the product details in the database.
* Replace or update product images in the file system or cloud storage.
* Reflect changes on the product detail and listing pages in real time.
* Display a success message to confirm the update.
* **Data Requirements:**
* Inputs:
* Product ID: Unique identifier for the product to be updated.
* Updated Product Details
* Outputs:
* Confirmation message: "Product updated successfully."
* Updated product details displayed on
* Validation rules:
* Product name and description must not be empty.
* Price and stock quantity must be positive numbers.
* Product images must be in supported formats

**Use Case 6: Find product**

1. **The scope of the work:**

* Frontend: Develop a user-friendly search bar and filtering interface.
* Backend: Implement product search logic, filtering mechanisms, and database queries.
* Testing: Validate accuracy, speed, and performance of search functionality

1. **The scope of the product:**

* Accurate and fast retrieval of relevant product information.
* User-friendly interaction with search, filtering, and sorting tools.

1. **Functional and Data Requirements:**

* **Functional Requirements:**
* Enter keywords in the search bar.
* Apply filters (e.g., category, price range, availability).
* Use sorting options (e.g., low-to-high price, relevance, or new arrivals).
* Click on a search result to view the product details.
* Accept the search query and filters.
* Query the database to find matching products.
* Return a list of products that match the criteria.
* Display search results dynamically.
* Update results when filters or sorting options are applied.
* Display a "No Results Found" message if no products match the search.
* **Data Requirements:**
* Inputs:
* Search Query: Keywords entered by the user.
* Filters
* Sorting Option: Selected sorting preference.
* Outputs:
* List of matching products.
* Total number of search results.
* "No Results Found" message if applicable.
* Validation rules:
* Ensure the search query is not empty.
* Validate that filtering and sorting parameters are within allowable ranges.

**Use Case 7: Register account**

1. **The scope of the work:**

* Frontend: Develop the registration form for inputting user details.
* Backend: Implement validation, secure storage, and communication with the database.
* Testing: Ensure the registration process works correctly for all valid inputs and handles errors gracefully.

1. **The scope of the product:**

* Proper validation and error handling of user input.
* User data is securely stored in the database.
* Users can easily access personalized services after registering.

1. **Functional and Data Requirements:**

* **Functional Requirments:**
* Navigate to the registration page.
* Agree to terms and conditions.
* Click the "Register" button to submit the form.
* Check if the email already exists in the database.
* Encrypt the password for secure storage.
* Store the user details in the database.
* Send a confirmation email or notification.
* **Data Requirements**
* **Inputs:**
* Full Name: User’s full name.
* Email Address: Unique identifier for the user.
* Password: User’s chosen password.
* Confirm Password: To ensure password correctness.
* Phone Number (optional): User’s contact number.
* Address (optional): User’s location.
* Agreement to Terms and Conditions.
* **Outputs:**
* Success Confirmation: "Account registered successfully."
* Error Messages
* **Validation rules:**
* Email must follow a valid format
* Password must meet strength requirements
* Confirm Password must match the Password.
* Email must not already exist in the system.

**Use Case 8: Update information**

1. **The scope of the work:**

* Frontend: Develop an intuitive "Update Information" page/form.
* Backend: Implement input validation, secure updates to the database, and error handling.
* Testing: Validate the update process, ensuring data accuracy and real-time feedback.

1. **The scope of the product:**

* Accurate validation of user input.
* Secure updates to the user database.
* Confirmation messages for successful updates or errors.

1. **Functional and Data Requirements:**

* **Functional Requirements:**
* Log in to their account.
* Navigate to the "Profile" or "Account Settings" page.
* Edit any of the following fields: Full name, Address, Phone number, Password
* Click "Save" or "Update" to confirm changes.
* Authenticate the user's session to ensure they are authorized.
* Retrieve and display the current user information.
* Validate the updated information
* Update the database with the new information.
* Reflect changes in the user’s account view.
* Display a success message.
* **Data Requirements:**
* Inputs:
* User ID: Unique identifier for the logged-in user.
* Updated Fields
* Outputs:
* Updated Information
* Confirmation Message
* Error Messages
* Validation rules:
* Email must follow a valid format and not already exist in the database.
* Phone number must be numeric and in the correct format.
* Password must meet the strength requirements (if updated) and match the confirmation field.
* Required fields cannot be empty.

**Use Case 9: Add order**

1. **The scope of the work:**

* Frontend: Develop a user-friendly checkout interface for submitting orders.
* Backend: Implement order processing logic, validation, inventory updates, and database storage.
* Integration: Integrate with payment gateways and order confirmation systems.
* Testing: Verify input validation, order placement, inventory updates, and payment handling.

1. **The scope of the product:**

* Proper validation of user details, payment methods, and stock availability.
* Automatic updates to inventory and order records.
* Users receive confirmation of a successful order.

1. **Functional and Data Requirements:**

* **Functional Requirements:**
* Navigate to the shopping cart.
* Review the cart and proceed to checkout.
* Enter or confirm shipping details (name, address, phone number).
* Select a payment method.
* Confirm the order.
* Authenticate the user’s session (if logged in).
* Validate the shopping cart.
* Validate shipping details and payment method.
* Deduct the ordered quantities from the inventory.
* Save the order details in the database.
* Generate an order confirmation summary.
* Send an order confirmation notification
* **Data Requirements:**
* **Inputs:**
* User ID (optional for guest checkout)
* Order Details
* Shipping Information
* Payment Method: Selected payment option
* **Outputs:**
* Order Summary
* Confirmation Message: "Order placed successfully."
* Error Messages
* **Validation rules:**
* Ensure all required fields (name, address, phone number) are provided.
* Verify product availability in inventory.
* Ensure payment method details are valid.
* Confirm the total order value matches the cart details.

**Use Case 10: Delete Order**

1. **The scope of the work:**

* Frontend: Develop an intuitive interface for deleting orders.
* Backend: Implement logic for validating order deletion, removing order records, and updating inventory.
* Testing: Verify successful deletion, proper inventory restoration, and error handling.

1. **The scope of the product:**

* Frontend: Develop an intuitive interface for deleting orders.
* Backend: Implement logic for validating order deletion, removing order records, and updating inventory.
* Testing: Verify successful deletion, proper inventory restoration, and error handling.

1. **Functional and Data Requirements:**

* **Functional Requirements:**
* Navigate to the "My Orders" or "Order Management" section.
* Select an order to delete.
* Click on the "Delete" or "Cancel Order" button.
* Access the "Order Management" dashboard.
* Search for and select an order to delete.
* Confirm deletion of the selected order.
* **Data Requirements:**
* **Inputs:**
* Access the "Order Management" dashboard.
* Search for and select an order to delete.
* Confirm deletion of the selected order.
* **Outputs:**
* Confirmation Messages
* Updated Data
* Error Messages
* **Validation rules:**
* Only allow users to delete orders with a "Pending" or "Unpaid" status.
* Ensure the Order ID exists in the database.
* Verify admin permissions for order deletion.

**Use Case 11: Update Order**

1. **The scope of the work:** Enable users to update existing orders with changes such as product quantity, shipping details, or order status.
2. **The scope of the product:**

* Allow customers to update their order quantity for furniture products.
* Enable administrators to change order details like product items, quantity, shipping address, and delivery status.
* Implement error handling for invalid inputs or constraints.
* Display a summary of the updated order before confirmation.
* Real-time synchronization with inventory for stock validation.
* Notification system for changes in order status (e.g., via email).

1. **Functional and Data Requirements:**

* **Functional Requirements:**
* Allow customers to update product quantity in an order.
* Allow administrators to update order details
* Validate new quantities against available stock.
* Display the updated order summary before confirmation.
* Notify customers/admins when an order is successfully updated.
* **Data Requirements:**

## NON-FUNCTIONAL REQUIREMENTS

1. **Performance**

* The system must ensure fast response times when users perform order update actions.
* Page load times should provide a smooth user experience.
* The system must handle a large number of concurrent users without interruptions.

1. **Scalability**

* The system should be capable of scaling flexibly to meet increasing user and order demands in the future.
* The database must support storing and processing a large volume of data without impacting overall performance.

1. **Reliability**

* The system must operate stably and continuously, minimizing downtime.
* Data integrity must be ensured, and no data should be lost during order updates.
* The system should be able to recover data efficiently in case of failures.

1. **Usability**

* The interface must be user-friendly, allowing users to update orders easily and intuitively.
* The website must work well across different devices and browsers.
* Clear notifications and instructions must be displayed to guide users when errors or issues occur.

1. **Security**

* User and order data must be securely stored and transmitted.
* Only authenticated users should be allowed to update orders.
* Proper access controls must be in place to prevent unauthorized changes or misuse of data.

1. **Maintainability**

* The system must be designed to be flexible and allow for easy upgrades or feature expansions.
* The codebase must be clear, well-documented, and maintainable for future developers.
* The system must support error logging and monitoring for quick issue resolution.

1. **Compatibility**

* The website must function smoothly on various web browsers and devices.
* The system should be compatible with external tools and services, such as payment gateways or notification services.

## Design

**Entiry-Relationship Diagram (ERD)**

**A diagram of a company

Description automatically generated**

**Figure 4: ERD**

**Use Case 1: Log in the system**

**A diagram of a login system

Description automatically generated**

**Use Case 2: Add product into cart**

**A diagram of a product

Description automatically generated**

**Use Case 3: Delete product from cart**

**A diagram of a product

Description automatically generated**

**Use Case 4: Update product from cart**

**A diagram of a process

Description automatically generated**

**Use Case 5: Update product in website**

**A diagram of a system

Description automatically generated**

**Use Case 6: Find product**

**A diagram of a computer process

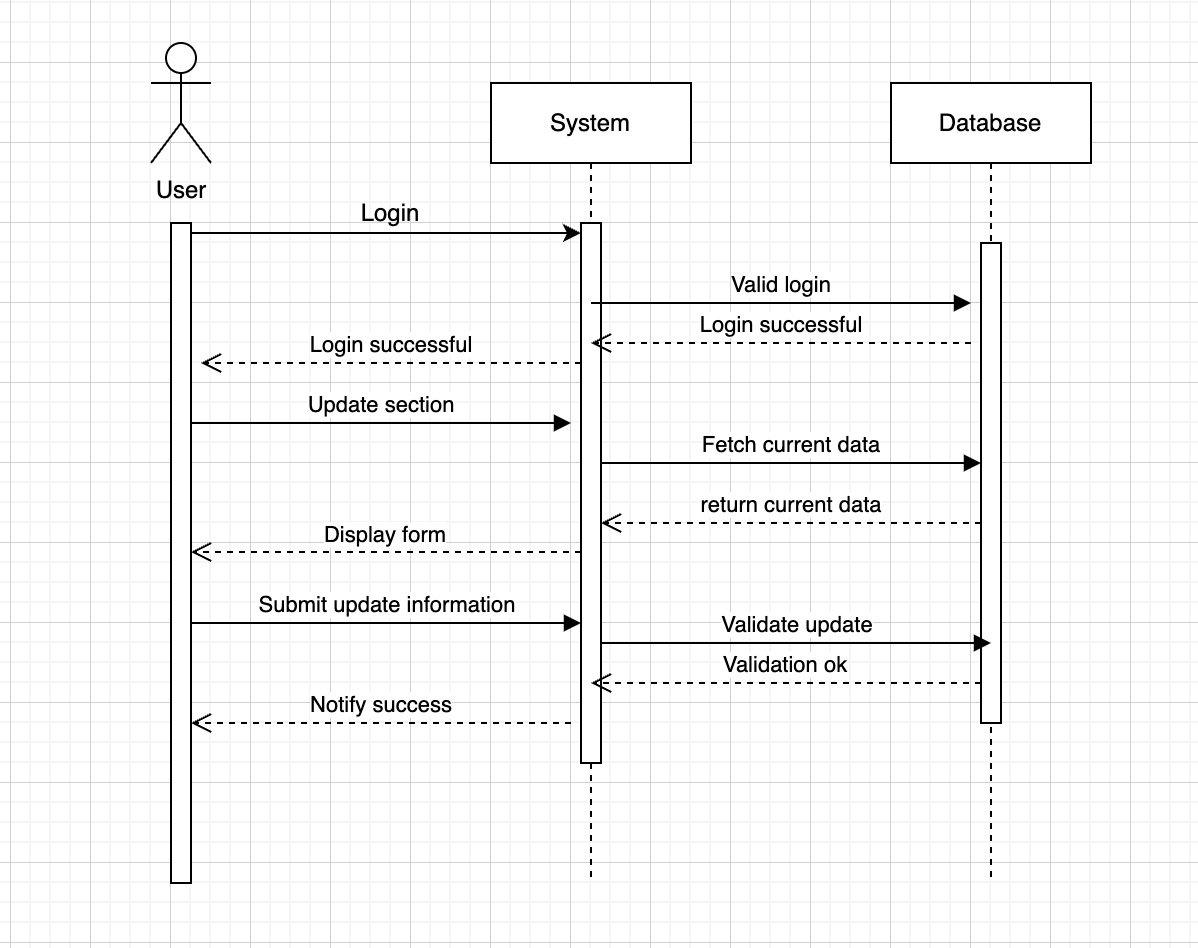
Description automatically generated**

**Use Case 7: Register account**

**A diagram of a system

Description automatically generated**

**Use Case 8: Update information**

****

**Use Case 9: Add order**

**A diagram of a product

Description automatically generated**

**Use Case 10: Delete Order**

**A diagram of a product

Description automatically generated**

**Use Case 11: Update Order**

**A diagram of a process

Description automatically generated**

# Django Framework

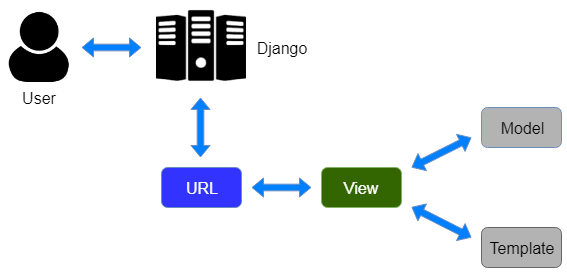
### MVT model

Django is based on MVT (Model-View-Template) architecture. MVT is a software design pattern for developing a web application.

MVT Structure has the following three parts:

* Model: The model is going to act as the interface of your data. It is responsible for maintaining data. It is the logical data structure behind the entire application and is represented by a database (generally relational databases such as MySql, Postgres).
* View: The View is the user interface — what you see in your browser when you render a website. It is represented by HTML/CSS/Javascript and Jinja files.
* Template: A template consists of static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted.

### MVT work flow

****

A user **requests** for a resource to the Django, Django works as a controller and check to the available resource in URL.

If URL maps**, a view** is called that interact with model and template, it renders a template.Django responds back to the user and sends **a template as a response**

### django.core.mail module

Although Python provides a mail sending interface via the smtplib module, Django provides a couple of light wrappers over it. These wrappers are provided to make sending email extra quick, to help test email sending during development, and to provide support for platforms that can’t use SMTP.

1. **Send\_mail()**

|  |
| --- |
| from django.core.mail import send\_mail  send\_mail(                  subject,                  message,                  sender,                  receiver,                  fail\_silently=False,              ) |

send\_mail(subject, message, from\_email, recipient\_list, fail\_silently=False, auth\_user=None, auth\_password=None, connection=None, html\_message=None)

In most cases, you can send email using django.core.mail.send\_mail().

The subject, message, from\_email and recipient\_list parameters are required.

1. **Setting**

|  |
| --- |
| EMAIL\_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'  EMAIL\_HOST = 'smtp.gmail.com'  EMAIL\_PORT = 587  EMAIL\_HOST\_USER = 'tqanhtkqn@gmail.com'  EMAIL\_HOST\_PASSWORD = 'bqvb neyd ogad deev'  EMAIL\_USE\_TLS = True  EMAIL\_USE\_SSL = False |

# IMPLEMENTATION

A screenshot of a computer

Description automatically generated

**Figure 5: Login page**

A screenshot of a computer

Description automatically generated

**Figure 6: User Interface**

A screenshot of a website

Description automatically generated

**Figure 7: Search Page Interface**

A screenshot of a computer

Description automatically generated

**Figure 8: Cart Interface**

**A screenshot of a computer

Description automatically generated**

**Figure 9: Homepage**

# DISCUSSION AND CONCLUSION

* During the implementation of the project, learning, understanding, and applying new technologies to the project was a challenge for the team.
* However, thanks to the efforts and hard work of each team member, we achieved several results, such as: building a basic interface for the Furniture Sales Website using HTML and CSS, optimizing website usability across multiple devices with the Bootstrap 5 framework, setting up a basic database for an e-commerce website, and using Django to implement website functionalities.
* The most satisfying feature is the shopping cart management, where users can easily add, update, and delete products. Additionally, displaying the product quantity and total price enhances user convenience.
* Despite these achievements, the team still has many shortcomings. For example, we were unable to apply Large Language Models (LLM) to the search functionality, did not allow users to apply discount codes, and some interface pages remain unfinished due to time constraints.
* The project has helped the team gain a deeper understanding of the process of developing an online sales website using Django and related technologies.

# REFERENCES

1. How to Build a Large Language Model from Scratch Using Python. Link: <https://www.freecodecamp.org/news/how-to-build-a-large-language-model-from-scratch-using-python/>
2. BootStrap5. Link: <https://getbootstrap.com/docs/5.3/getting-started/introduction/>
3. What is the use of {% load static %}?. Link: <https://stackoverflow.com/questions/59308372/what-is-the-use-of-load-static>
4. What is a CSRF token? What is its importance and how does it work?. Link: <https://stackoverflow.com/questions/5207160/what-is-a-csrf-token-what-is-its-importance-and-how-does-it-work>